IN-60

114118

P- 12

NASA
Technical Memorandum 105795

AVSCOM
Technical Report 92-C-014

Implementation of an Intelligent Control System

D.L. Simon
Propulsion Directorate
U.S. Army Aviation Systems Command
Lewis Research Center
Cleveland, Ohio

and

E. Wong and J.L. Musgrave
Lewis Research Center
Cleveland, Ohio

Prepared for the
1992 Advanced Earth-to-Orbit Propulsion Technology Conference
sponsored by Marshall Space Flight Center
Huntsville, Alabama, May 19–21, 1992

NASA

IMPLEMENTATION OF AN INTELLIGENT CONTROL SYSTEM

D. L. Simon
US Army Aviation Systems Command
Propulsion Directorate
Lewis Research Center
Cleveland, Ohio 44135

E. Wong and J. L. Musgrave
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

ABSTRACT

A laboratory testbed facility which has been constructed at the NASA Lewis Research Center for the development of an Intelligent Control System (ICS) for reusable rocket engines is described. The framework of the ICS consists of a hierarchy of various control and diagnostic functions. The traditional high speed, closed-loop controller resides at the lowest level of the ICS hierarchy. Above this level resides the diagnostic functions which identify engine faults. The ICS top level consists of the coordination function which manages the interaction between an expert system and a traditional control system. The purpose of the testbed is to demonstrate the feasibility of the ICS concept by implementing the ICS as the primary controller in a simulation of the Space Shuttle Main Engine (SSME). The functions of the ICS which are implemented in the testbed are: an SSME dynamic simulation with selected fault mode models, a reconfigurable controller, a neural network for sensor validation, a model-based failure detection algorithm, a rule based failure detection algorithm, a diagnostic expert system, an intelligent coordinator, and a user interface which provides a graphical representation of the events occurring within the testbed. The diverse nature of the ICS has led to the development of a distributed architecture consisting of specialized hardware and software for the implementation of the various functions. This testbed is made up of five different computer systems. These individual computers are discussed along with the schemes used to implement the various ICS components. The communication between computers and the timing and synchronization between components are also addressed.

INTRODUCTION

An Intelligent Control System (ICS) is under development at the NASA Lewis Research Center with the primary objective of improving the operability and maintainability of an earth-to-orbit propulsion system. In particular, the focus of this technology program is the Space Shuttle Main Engine (SSME). The SSME is the first large scale, liquid propellant reusable rocket engine developed from a long line of expendable rocket technology. High thrust levels required for a typical SSME mission have pushed the notion of "reusability" into the future by requiring post-flight checkout of numerous major engine components. Merrill and Lorenzo have proposed a framework outlining specific functionalities to improve the durability of the SSME which include active control of key engine parameters, real time diagnostics, and life extending control [1].

An aggressive technology program has been in progress at LeRC since 1989 with proof-of-concept targeted for 1995 using the framework proposed by Merrill and Lorenzo as a road map for technology development. Towards this end, a simulation testbed has been assembled providing a flexible vehicle for performing the research necessary for development of ICS functionalities. In addition, the simulation facility provides a suitable environment for examining a variety of fault scenarios and demonstrating the capabilities of the ICS. Effort in the present program is not limited to the synthesis of diagnostic and control algorithms. Development of special purpose hardware for performing various ICS functions is an important aspect of the present program. This paper describes the testbed hardware, the testbed software, and the communication and handshaking between individual testbed components. It will also address future plans for upgrading the present testbed. Before beginning a discussion of the implementation of the ICS, a brief overview of the basic capabilities included in the system is put forth in the context of the functional framework.

INTELLIGENT CONTROL SYSTEM FRAMEWORK

The ICS functional framework in Figure 1 contains many elements of the original framework of Merrill and Lorenzo but is slightly narrower in scope since prognostic and life extending capabilities are not considered for the near term demonstration. The primary objective of the program is the successful integration of on-line diagnostics with accommodation actions using reconfigurable controls for selected fault modes of rocket engines [2].

The real-time diagnostic system in Figure 1 is composed of sensor validation, model based failure detection, rule based failure detection, and ReREDS (Reusable Rocket Engine Diagnostic System) a diagnostic system developed over the past two years through a contract with System Control Technology (SCT) and Aerojet. Each subsystem is well suited to identifying a certain class of engine faults. In this way, they compliment one another through their respective strengths leaving the final determination of an engine fault to a higher level diagnostic expert system. The diagnostic expert system is based on heuristic rules that determine which fault has actually occurred based upon the data provided by the various subsystems. The distributed nature of the diagnostic system allows a divide and conquer strategy with information compression performed at lower levels in the hierarchy to relieve the computational burden in the diagnostic expert system thereby facilitating real-time operation [3]. The engine level coordinator makes alterations to the controller using engine status information generated by the diagnostic system, and propulsion requirements passed down by the propulsion level coordinator which receives thrust vector commands from the flight controller. Ultimately, the engine level coordinator must satisfy minimum thrust requirements while minimizing further component degradation and accommodating failed or degraded engine hardware. The reconfigurable controller takes requests generated by the coordinator, makes the changes gradually thereby minimizing engine transients, and computes the valve positions to achieve the requested behavior from the engine. The control is robust to variations in engines and degradations in components. In addition the control takes advantage of available hardware redundancy in the propulsion system to maximize fault tolerance.

THE INTELLIGENT CONTROL SYSTEM TESTBED

As described previously, the intelligent control system framework consists of various control and diagnostic functionalities. This diversity has led to a distributed testbed architecture consisting of specialized computers and software for the implementation of the various functions. Having specialized computers has simplified the development process and has provided the necessary capabilities required by the different functionalities. However, a distributed implementation does pose some problems. It leads to increases in communication and in complexity. Also some of the computers do not have the capability to run in real-time or have limited I/O capabilities. The present testbed is a non real-time implementation which is running 10 times slower than real-time. The main purpose for the present testbed is to demonstrate the feasibility of the ICS concept. In the future, hardware and software for the real-time implementation of specialized functions such as neural networks and expert systems should become more readily available. The present ICS testbed will be upgraded to a real-time system by taking advantage of such advances.

Figure 2 shows the Intelligent Control System testbed as well as the functionality implemented on each to the five computers. It should be noted that ReREDS is not incorporated into the present version of the testbed. However, the testbed will eventually allow the entire ICS functional framework shown in Figure 1 to be implemented. The five computers included in the present testbed are the simulation computer, controls computer, neural network computer, diagnostic computer, and a computer used to implement the graphical user interface. It should be noted that ReREDS has not yet been incorporated into the testbed. In the following sections the individual computers will be discussed in detail along with the communication and synchronization between them.

TESTBED COMPONENTS

Simulation Computer

A non-linear simulation of the SSME including fault mode models is implemented on the simulation computer. The simulation computer is specifically designed for the real time implementation of continuous, dynamic systems. The system consists of an Applied Dynamics System 100 simulation computer, analog and digital I/O hardware for running hardware-in-the-loop simulations, and a Vaxstation II front end. The simulation computer is a 64-bit floating point multiprocessor which is optimized for numerical integration. The I/O facilities provide analog communications to allow the SSME simulation to communicate with the rest of the ICS testbed. The Vaxstation II front end computer is where all of the program development occurs such as editing, compiling, and linking the simulation. The Vaxstation also runs data collection and graphical display utilities to allow the user to monitor and evaluate the simulation performance. The models implemented on the simulation computer are coded in ADSIM, a proprietary programming language of Applied Dynamics. The simulation is updated via an internal timer at a user specified update rate.

Controls Computer

The SSME multivariable reconfigurable controller, the intelligent coordinator, and the model based failure detection algorithm are coded in FORTRAN and are implemented on the controls computer. The controls computer system is known as the Control, Interface, and Monitoring (CIM) Unit. This unit was fabricated in-house at NASA Lewis for the implementation and evaluation of advanced digital control algorithms with hardware in the loop [4]. It is intended for use in both simulation facilities and engine test facilities, and is therefore implemented in a portable equipment rack. The controls computer consists of a microcomputer running a real time operating system, interface hardware for connecting to an engine simulation or actual engine hardware, and monitoring hardware and software to verify that the control is performing properly. The real time capabilities of the controls computer make it well suited for running the reconfigurable controller as well as the intelligent coordinator and the model based failure detection algorithm.

The microcomputer consists of a 33 MHz 80486 single board computer, analog and discrete I/O boards, an ethernet controller board, and disk drives with their associated controller boards. The circuit boards are mounted in an industry standard Multibus I chassis. The microcomputer runs the iRMX real time operating system. This operating system provides the services needed to construct a real time executive to schedule the execution of the multivariable control algorithm, the model-based failure detection algorithm, I/O routines, and the data collection software. The executive is coded in PL/M and updated by a timer generated interrupt at a user specified rate. Analog I/O is used for the communication from the controls computer to the simulation computer and the neural network PC. A bi-directional digital interface is used to implement the communication between the controls computer and the diagnostic computer. The executive schedules communication to the user interface over an ethernet link and the internal data collection utility as lower level tasks.

The purpose of the interface hardware is to route signals throughout the controls computer, to connect the controls computer to external devices, and to buffer those signals if desired. Cables carrying analog signals which interface to both the AD100 and the neural net PC are terminated at the controls computer base connectors. A patch panel is available to control the routing of analog signals throughout the system and to allow quick changes in the configuration. The controls computer can support up to 32 analog outputs and 64 analog inputs. The front panel of the controls computer has 32 lights and 24 switches that can be configured by the user to perform various functions. The lights are used to indicate the status of the software executing on the processor and the switches are used by the operator to change modes of operation.

The monitoring capabilities of the controls computer allow the user to observe analog signals that are sent to and from the system, as well as record variables within the control algorithms during execution. A data acquisition system monitors the control computer I/O and allows the operator to display any desired signal or signals in actual voltages or engineering units. The Microcomputer Interactive Data System (MINDS) program runs in the background on the microcomputer CPU [5]. The MINDS program has both steady-state and transient data collection capabilities and can access any variable in the control algorithm. MINDS allows collected data to be stored to hard disk or displayed to the terminal.

PC Neural Network Implementation System

The sensor validation neural network is incorporated into the ICS testbed through the PC neural network implementation system shown in Figure 3. The hardware consists of an 80286 PC/AT cabled to a separate expansion chassis which essentially extends the PC bus and provides additional expansion slots. This chassis connection was necessary, because of the large number of peripheral boards required. Installed onto this setup is an ANZA Plus neural processor board from Hecht-Nielson Corporation. This board does not actually contain true neural network devices; such neural net hardware is just now beginning to appear on the market and does not yet possess the necessary sophistication. Rather, the ANZA Plus is an accelerator board which utilizes the Intel i860 reduced instruction set computing (RISC) processor chip. This processor has the ability to multiply and accumulate in a single cycle, thus allowing it to speed up interconnect calculations and other neural computations. System communication was provided by installing analog I/O hardware. This hardware provides the system with an I/O capability consisting of 16 inputs and 12 analog outputs. Since the system interface consists of analog signals, the neural network implementation has the appearance of a true analog neural net to the other components of the testbed.

The software developed for the PC neural network implementation system consists of a main executive program, which schedules the operation of the ANZA neural processor and the I/O boards, as well as various procedures grouped into functional modules. This main executive performs the required declarations and definitions and initializes the hardware. It then schedules the conversion of analog signals into neural net inputs, the download of input data to the ANZA board, the iteration of network computations on the ANZA board, the conversion of computed neural network outputs into analog signals and the updating of the display screen. These operations are performed by the executive through procedure calls. These procedures are grouped by function into the following modules: I/O procedures, neural net procedures, display procedures and timing procedures.

The I/O routines provide an interface to the A/D and D/A boards. They initialize conversions, access control and data registers on the boards and perform the appropriate scaling of input and output data. The display routines initialize the screen and provide continual display updates of the network inputs and outputs in the form of a graph. The timing routines provide a means of measuring the cycle update times for benchmarking purposes, even though the system is configured to execute as fast as possible and is not synchronized to any other components of the testbed. The neural network procedures are used to initialize the ANZA board, initialize a network structure, load pre-trained weights, put input data into the board, initiate net iterations and get computed outputs from the board. These procedures interface with the ANZA processor board via the HNC User Interface Subroutine Library (UISL) which is a set of commands used to control processor operations. The ANZA board was not designed to implement on-line neural network operating within an environment of continuously changing inputs. The present neural network implementation requires approximately 50 milliseconds for each update. Much of this delay can be attributed to the time required to download new data and then read the corresponding outputs from the ANZA board. This update rate is sufficient for the present scaled time implementation of the ICS but improved hardware will be needed in the future to allow us to implement neural networks in real time.

Diagnostic Computer

A commercially available expert system shell, G2™ from Gensym, was chosen to run the diagnostic expert system and the component failure detection algorithm. G2 has been implemented on a Vaxstation 3500 computer to incorporate it into the ICS testbed. The user defines objects to represent various aspects of the environment using an object oriented approach. After reviewing currently available software shells in the artificial intelligence field, the G2 expert system was selected based on three main criteria: (1) its flexibility in expanding the knowledge base; (2) its capability to perform numerical simulation; and (3) its capability to perform on-line input/output operations [6].

The G2 software package provides a rule-based inference engine which allows for the future expansion of the knowledge base. This is important for the ICS program since it is a research project where the future addition of rules is a certainty. Another important feature of G2 is its integrated simulation capability. A numerical model can be programmed in the simulator to provide a possible source of values for variables. This simulation facility provides the potential of testing the knowledge base under development without connecting to an actual system. Along with the expert system shell, an interface called GSI™ can be set up to provide input/output data. GSI allows the inference engine to operate on continuously changing input.

One major limitation of G2 is its sampling rate. The current defined minimum sample interval is one second. This is not fast enough for rocket engine fault detection and is one of the key reasons why the present ICS demonstration is not running in real time. We expect that G2 will let us gain insight into how the real-time decision making process should be handled. In the future as we move towards a real-time testbed implementation we will need improved software to allow us to implement the diagnostic expert system in real-time.

Graphical User Interface and ReREDS Computer

The Graphical User Interface (GUI) and ReREDS are both implemented on a TI Explorer II Lx +. The GUI was developed to allow the ICS to be monitored during operation. The GUI permits users to observe the ICS in real-time operation as it accommodates faults in components, sensors, and actuators, using a collection of screens designed to provide a clear illustration through plots, text, and animation of the entire process. The TI Explorer II Lx + is well suited for the development of knowledge based systems and is well suited for the ReREDS application which is coded in LISP. However its object oriented programming capacity makes the development of graphical user interfaces straight forward. The GUI is a full-color, object-oriented system consisting of a set of screens arranged hierarchically. Each screen consists of three windows: a mouse-sensitive graphical display window containing a diagram of a component or system, a plotting window depicting time responses of key variables associated with that component or system, and an interactive type-out window displaying messages and allowing the user to enter commands. Figure 4 shows an example screen. The top window contains a view of the space shuttle main engine composed of selectable objects, the window on the lower left displays messages, and that on the lower right displays plots. The CIM Unit passes data and coded messages to the TI Explorer over an Ethernet connection. The GUI plots time responses of important variables and indicates engine faults to the user through messages in the type-out window and by causing failed mouse-selectable components to flash. The user may bring up more detailed screens by clicking the mouse on the objects. Because of the modular, object-oriented nature of the GUI, the creation of additional screens is simple and quick. Thus appropriate screens can be added easily as more fault modes are incorporated into the testbed system.

SYSTEM COMMUNICATIONS AND SYNCHRONIZATION

The implementation of a system distributed among several different computers naturally yields a rather complex communication system. The Intelligent Control System implementation consists of analog, digital, and network communications as shown in Figure 2. From the diagram we can see that the controls computer is the hub of testbed communications. The reason for this is twofold. The controls computer is running the multivariable controller, the engine level coordinator, and the model based failure detection algorithm which are central components of the ICS framework. The controls computer also supports a wider array of communication options than most of the computers in the testbed. Therefore two computers that needed to share data but did not have a compatible communication link resolved the problem by transferring data through the controls computer. For example sensor validation information calculated in the PC neural net implementation system is passed enroute to the diagnostic expert system on the diagnostic computer through the controls computer.

All I/O between the engine simulation, controls computer, and sensor validation neural network is analog. This implementation is straight forward and is representative of an actual engine where the sensor outputs and actuator inputs would be in the form of analog signals. Each of the three computers use 12 bit A/D and D/A converters. Incoming analog values are scaled to their corresponding values in engineering units. Similarly, outgoing signals are scaled before being sent to the D/A converters. The neural net PC is running asynchronously from the other computers within the testbed. It presently requires 50 milliseconds to update, most of which can be attributed to the required transfer of data in and out of the neural network accelerator board on each update. The controls computer uses a multitasking scheme to schedule the execution of its software, communication, and data collection responsibilities. The multivariable controller and the model based failure detector are designed to update in real time at rates of 10 milliseconds and 40 milliseconds respectively. The scaled versions run at 100 and 400 milliseconds. There are five different tasks scheduled to operate on the controls computer. These tasks have been implemented so that the fastest and most urgent tasks have the highest priority. This means that they can temporarily suspend all lower priority tasks until they have completed operation. The five tasks are as follows: The highest priority task executes the multivariable control, the engine level coordinator, and handles the analog I/O as well as communication over the DR11 interface. The second task executes the model based failure detection algorithm. The third task performs MINDS transient data sampling. The fourth task performs the transfer of data to the user interface over the ethernet connection. The fifth and lowest priority task is the MINDS utility which interprets user input supplied from the keyboard. Figure 5 shows a diagram for the five tasks arranged from top to bottom in the order of their priority. This figure is intended for illustration purposes and does not accurately represent the relative execution times for the five tasks.

The interface between the controls computer and the diagnostic computer is implemented through a DR11 interface. This is a bi-directional parallel digital interface which transfers data in 16 bit words. Figure 6 shows the interaction between the controls computer and the G2 application running on the diagnostic computer. The diagnostic computer sets a flag when it wishes to send or receive data and then waits for the controls computer to respond. The controls computer polls the same flag on each update. When it detects that the flag has been set, it initiates the data transfer. This polling scheme insures that the controls computer, which is operating at a much faster update rate, is never idle and waiting for the diagnostic computer to respond.

Information is transferred from the controls computer to the user interface over an ethernet connection. The TCP/IP protocol is used to develop the software to transfer packets of floating point data. As shown in

Figure 5, the task on the controls computer which handles this communication is assigned a relatively low priority. This is because of the delay and non-deterministic behavior associated with data transmission over ethernet.

SUMMARY

A testbed which has been developed at the NASA Lewis Research Center for the implementation of an Intelligent Control System (ICS) for earth-to-orbit propulsion systems is described. The diverse nature of the ICS has led to the development of a distributed testbed consisting of specialized hardware and software. The present testbed is being used to demonstrate the operation of the ICS in scaled time. Future plans include upgrading the hardware and software to obtain real time operation and also expanding the testbed to include additional functionality.

REFERENCES

1. Merrill, W.C. and Lorenzo, C.F., "A Reusable Rocket Engine Intelligent Control," AIAA-88-3114, 1988 AIAA Joint Propulsion Conference, Boston, MA., July 11-13, 1988.

2. Merrill, W.C., Musgrave, J.L., and Guo, T.H., "Integrated Health Monitoring and Controls for Rocket Engines," 921031, 1992 SAE Aerospace Atlantic Conference, Dayton, Oh., April 7-10, 1992.

3. Guo, T.H. and Merrill, W.C., "A Framework for Real-Time Rocket Engine Diagnostics," 1990 Conference on Advanced Earth-to-Orbit Propulsion Technology, Marshall Space Flight Center, Huntsville, Al., May 15-17, 1990.

4. DeLaat, J.C., Soeder, J.F., "Design of a Microprocessor-Based Control, Interface, and Monitoring (CIM) Unit for Turbine Engine Control Research", NASA TM-83433, 1983.

5. Soeder, J.F., "MINDS: A Microcomputer Interactive Data System for 8086-Based Controllers," NASA TP-2378, January 1985.

6. Guo, T.H., "An SSME High Pressure Oxidizer Turbopump Diagnostic System Using G2™ Real-Time Expert System," Health Monitoring Conference for Space Propulsion Systems, Cincinnati, Oh., November 13-14, 1991.

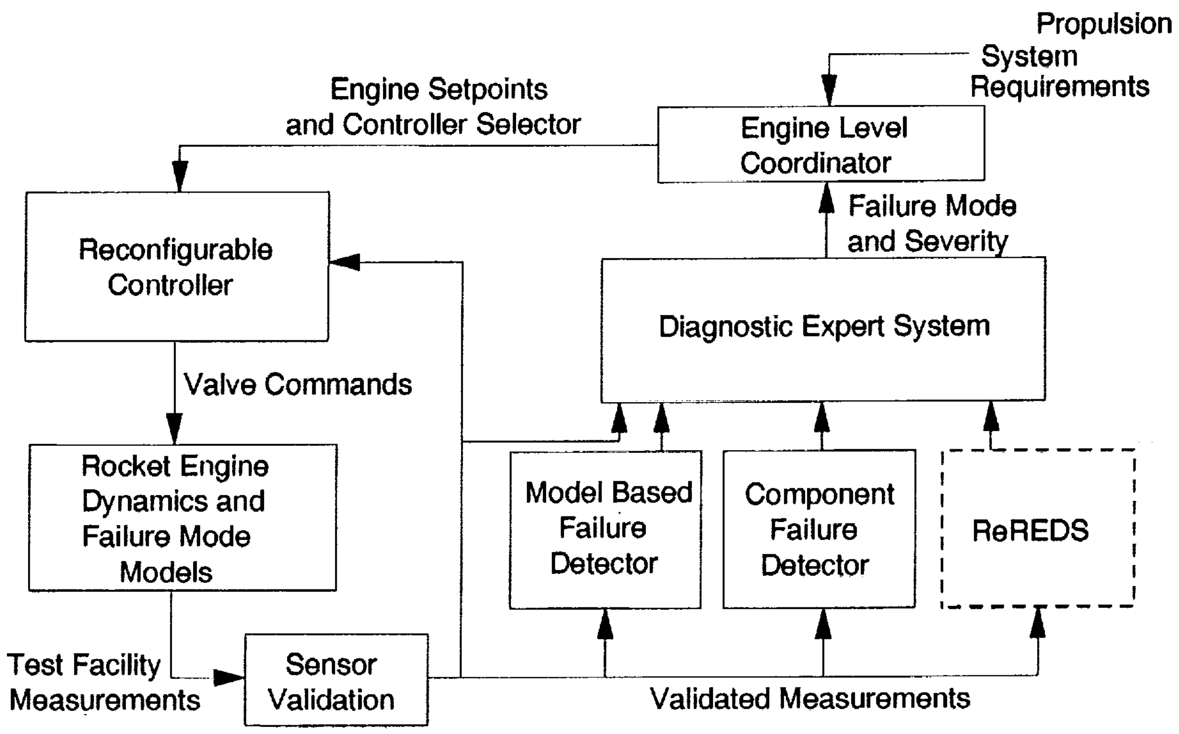

Propulsion System Requirements
Engine Setpoints and Controller Selector
Engine Level Coordinator
Failure Mode and Severity
Reconfigurable Controller
Diagnostic Expert System
Valve Commands
Rocket Engine Dynamics and Failure Mode Models
Model Based Failure Detector
Component Failure Detector
ReREDS
Test Facility Measurements
Sensor Validation
Validated Measurements

Figure 1. Intelligent Control System Functional Framework

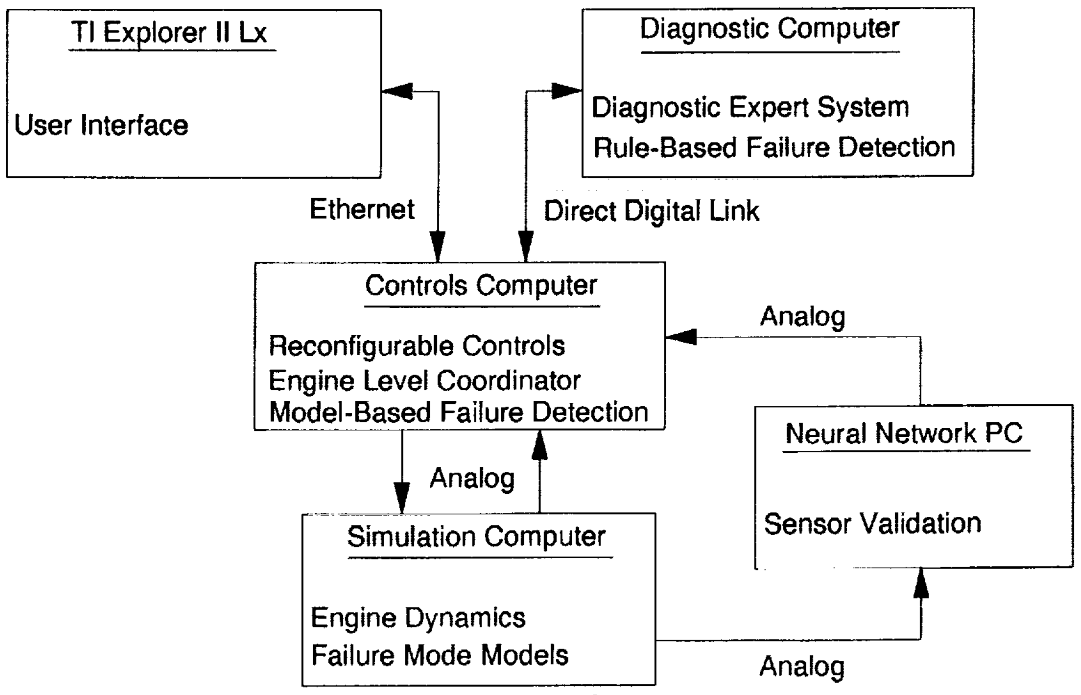


Figure 2. Intelligent Control System Simulation Testbed

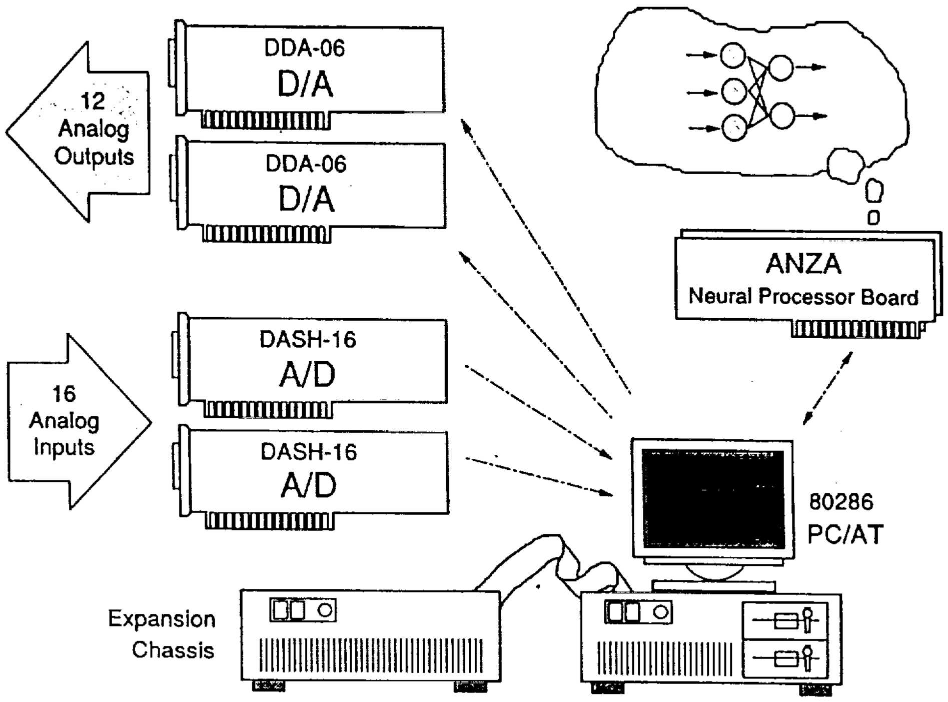
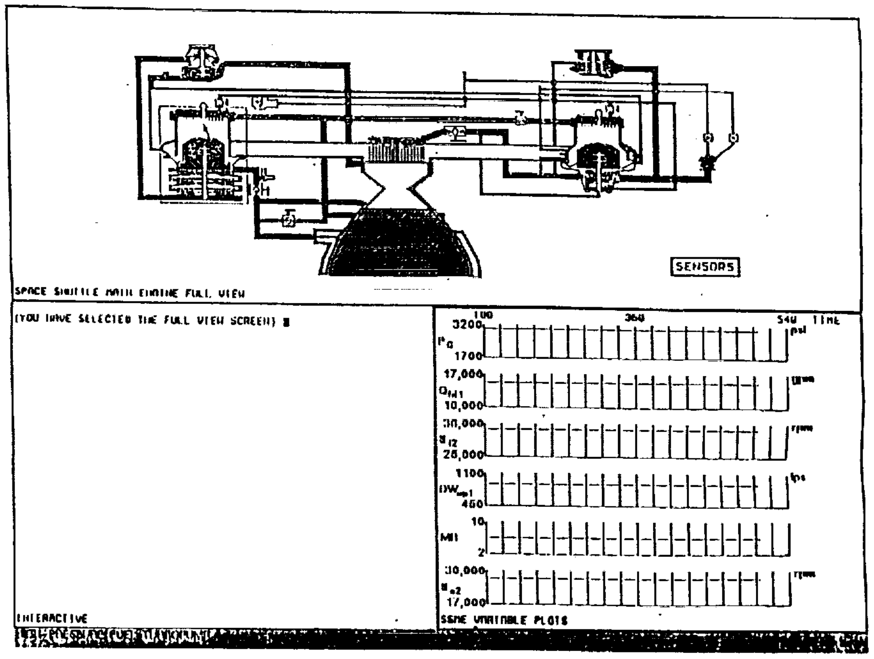


Figure 3. Neural Network Implementation Hardware


SPACE SHUTTLE MAIN ENGINE FULL VIEW

(YOU HAVE SELECTED THE FULL VIEW SCREEN)

INTERACTIVE

SSME VARIABLE PLOTS

Figure 4. Graphical User Interface Screen

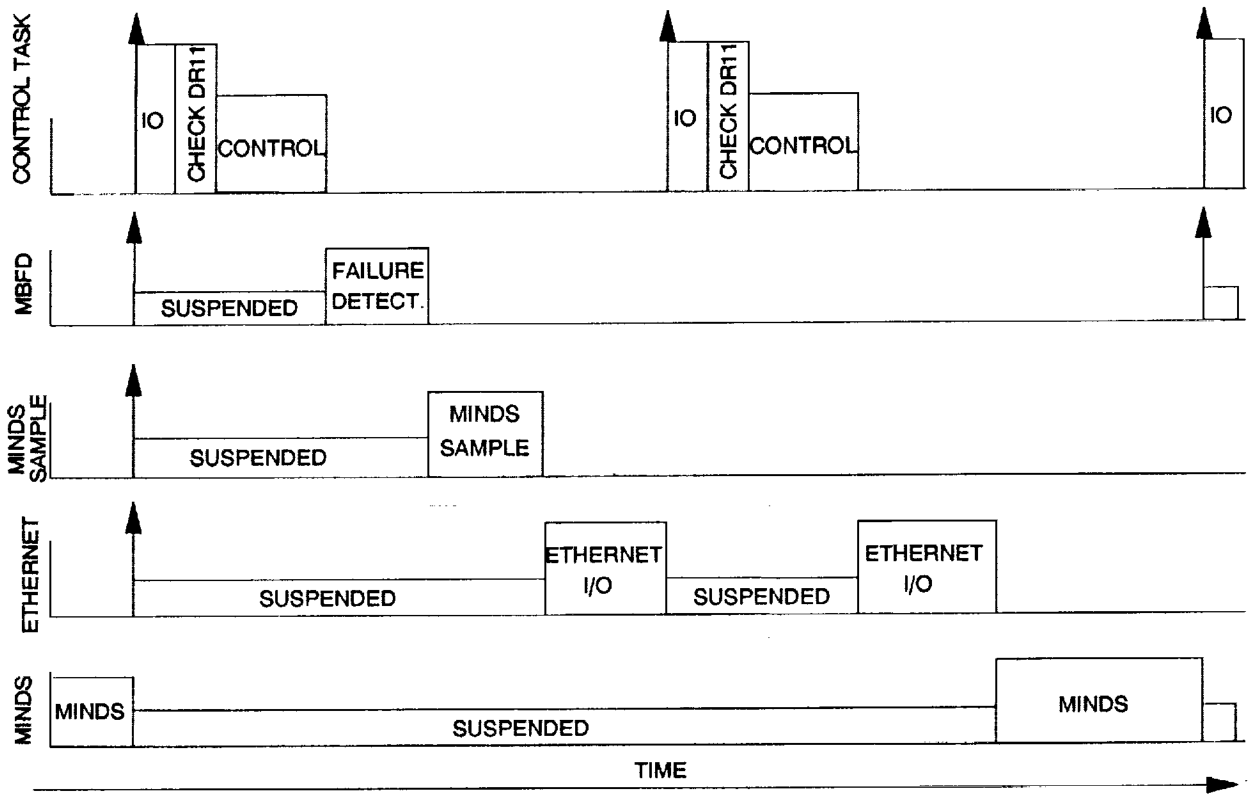


Figure 5. Controls Computer Task Scheduling

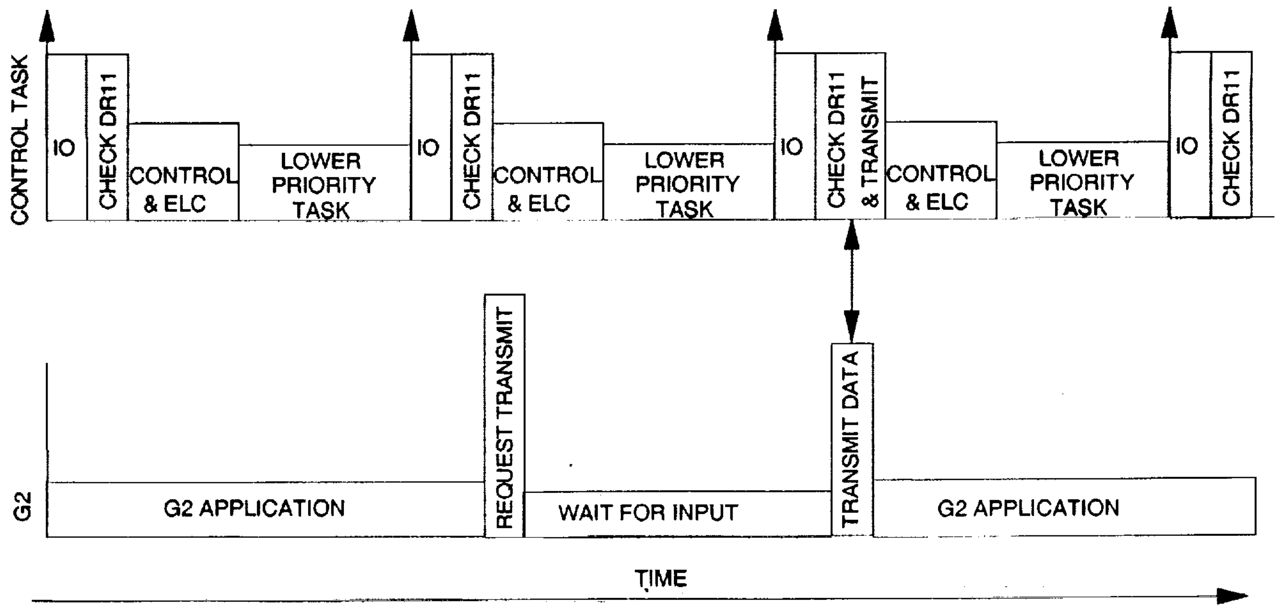


Figure 6. Data Transmission Over DR11 Interface

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE May 1992	3. REPORT TYPE AND DATES COVERED Technical Memorandum

4. TITLE AND SUBTITLE Implementation of an Intelligent Control System	5. FUNDING NUMBERS
6. AUTHOR(S) D.L. Simon, E. Wong, and J.L. Musgrave	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) NASA Lewis Research Center Cleveland, Ohio 44135–3191 and Propulsion Directorate U.S. Army Aviation Systems Command Cleveland, Ohio 44135–3191	8. PERFORMING ORGANIZATION REPORT NUMBER E–7225
9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, D.C. 20546–0001 and U.S. Army Aviation Systems Command St. Louis, Mo. 63120–1798	10. SPONSORING/MONITORING AGENCY REPORT NUMBER NASA TM–105795 AVSCOM TR–92–C–014

11. SUPPLEMENTARY NOTES

Prepared for the 1992 Advanced Earth-to-Orbit Propulsion Technology Conference sponsored by Marshall Space Flight Center, Huntsville, Alabama, May 19–21, 1992. D.L. Simon, Propulsion Directorate, U.S. Army Aviation Systems Command; and E. Wong and J.L. Musgrave, NASA Lewis Research Center, Cleveland, Ohio. Responsible person, D.L. Simon, (216) 433–3740.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified–Unlimited Subject Category 60	12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

A laboratory testbed facility which has been constructed at the NASA Lewis Research Center for the development of an Intelligent Control System (ICS) for reusable rocket engines is described. The framework of the ICS consists of a hierarchy of various control and diagnostic functions. The traditional high speed, closed-loop controller resides at the lowest level of the ICS hierarchy. Above this level resides the diagnostic functions which identify engine faults. The ICS top level consists of the coordination function which manages the interaction between an expert system and a traditional control system. The purpose of the testbed is to demonstrate the feasibility of the ICS concept by implementing the ICS as the primary controller in a simulation of the Space Shuttle Main Engine (SSME). The functions of the ICS which are implemented in the testbed are: an SSME dynamic simulation with selected fault mode models, a reconfigurable controller, a neural network for sensor validation, a model-based failure detection algorithm, a rule based failure detection algorithm, a diagnostic expert system, an intelligent coordinator, and a user interface which provides a graphical representation of the events occurring within the testbed. The diverse nature of the ICS has led to the development of a distributed architecture consisting of specialized hardware and software for the implementation of the various functions. This testbed is made up of five different computer systems. These individual computers are discussed along with the schemes used to implement the various ICS components. The communication between computers and the timing and synchronization between components are also addressed.

14. SUBJECT TERMS Controls; Diagnostics; Expert systems; Computerized simulation	15. NUMBER OF PAGES 12
	16. PRICE CODE A03

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT